第1章

“FPGAマイコン”最新事情

タダ同然で使えるCPUコアを活用しよう

浅井 剛

ここでは，FPGA向けに提供されるソフト・マクロのCPUコアの普及状況を説明する．現在では，ほとんどのFPGAベンダがタダまたはタダ同然の価格でCPUコアを提供している．FPGAの集積度が上がり，コストが下がったことでますます使いやすくなってきている．また，CPU向けのソフトウェア開発ではC言語が広く用いられている．加えて最近ではCベース言語によるハードウェア設計も注目を集めている．“FPGAマイコン”を使う設計におけるCベース言語の可能性についても考える．

（編集部）

米国Altera社がソフト・マクロのCPUである「Nios」を発表したのは，2000年6月のことでした．それから6年以上が経過し，現在ではFPGAベンダがみずから提供するソフト・マクロのCPUが増えています（**表1**）．

1．FPGAベンダが提供するソフトCPU

FPGAベンダが提供するCPUコアの特徴は，自社のデバイスに最適化していることでしょう．汎用のIPコアと比べると，性能や論理規模の面で有利です．FPGAベンダにとっては，自社製品の価値を高めるという目的があるため，中途半端な機能や性能のコアを提供することは考えられません．またIPコアの販売をビジネスとして考える必要がないので，コスト・パフォーマンスも優れています（**表1**）．

● Nios II

FPGAベンダが提供するCPUコアの代表的なものの一つが，Altera社の「Nios II」です．

初代の「Nios」には，16ビット版と32ビット版がありました．現在の「Nios II」は32ビット版だけです．Nios IIには排他制御を行うMutexコアがサポートされているため，マルチプロセッサ構成も実現しやすくなりました．ハードウェア，ソフトウェア，開発ツールすべての面でアップデートを重ねながら現在に至っています．

一方，Nios IIの発表に伴いNiosのアップデートは終了しました．このため，Niosでは最新のFPGAファミリがサポートされていません．Niosが使われ始めたころ，ディスコン（製造中止）対策として活用する事例が報告されていました．設計データを残せても，搭載できるFPGAが入手できなくなればディスコンといっしょです．機能のアップデートがなくなるとしても，新しいFPGAへの対応だけは続けてほしいところです．

● MicroBlaze，PicoBlaze

米国Xilinx社が提供するCPUコアで，32ビット版の「MicroBlaze」と8ビット版の「PicoBlaze」があります．

MicroBlazeでは，OPB（On-Chip Peripheral Bus）に米国IBM社のCoreConnectというバス・プロトコルを採用しています．Virtex-4などにハード・マクロで搭載されるCPUのPowerPCとの組み合わせも可能です．

PicoBlazeはライセンス，ロイヤリティともに無償で提供されています．ユーザ登録さえすればだれでもダウンロードできます．発表当初はVHDLだけの提供でしたが，最近はVerilog HDLも提供されているようです．

KeyWord ソフト・マクロ，CPUコア，Nios II，MicroBlaze，PicoBlaze，CoreMP7，LatticeMico8，LatticeMico32，SOC，C言語，SystemC，SpecC

表1
FPGAベンダが提供するソフト・マクロのCPU
ほとんどのFPGAベンダがCPUコアを提供している．米国Xilinx社や米国QuickLogic社は，ハード・マクロでCPUを搭載するFPGAも発売している．

メーカ	名　称	CPU動作周波数	適用デバイス	標準提供ペリフェラル
Actel社	CoreMP7	90.7MHz	M7 Fusion M7 ProASIC3 M7 ProASIC3/E	AHB-Liteインターフェース APBインターフェース AHB-APBブリッジ メモリ・コントローラ 割り込みコントローラ タイマ，SIO ほか
Altera社	Nios	125MHz	Stratix Cyclone APEX/APEX II ACEX FLEX	DMAコントローラ UART，SPI，PIO メモリ・コントローラ タイマ ほか
	Nios II	250MHz	Stratix Stratix II Cyclone Cyclone II	上記に加え MMU（予定）
Lattice Semiconductor社	LatticeMico8	77.1MHz	LatticeECP LatticeEC LatticeXP MachXO	UART
	LatticeMico32	110MHz	LatticeECP2 LatticeECP LatticeEC LatticeSC LatticeXP	SRAMコントローラ メモリ・コントローラ タイマ，GPIO DMAコントローラ I^2Cインターフェース SPI，UART
Xilinx社	MicroBlaze	200MHz	Spartan-3/E Spartan-II/E Virtex-E Virtex-II/Pro Virtex-4	UART GPIO メモリ・コントローラ タイマ アービタ
	PicoBlaze （旧KCPSM）	200MHz	Spartan-3 Virtex-II/Pro Virtex-4	GPIO ほか

● CoreMP7

米国Actel社の「CoreMP7」は，英国ARM社のARM7コアを最適化したものです．

オンチップ・バスとしては標準的なAHB（Advanced High-performance Bus）とAPB（Advanced Peripheral Bus）をサポートします．また，メモリ・コントローラや割り込みコントローラなど，ARMの性能を引き出すために必要なモジュールをすべて備えています．JTAGデバッグ・インターフェースを使ったICE（in-circuit emulator）によるデバッグも可能です．

CoreMP7を使用するにあたって，ライセンス料やロイヤリティについて，FPGAユーザは考える必要がありません．

● LatticeMico8，LatticeMico32

米国Lattice Semiconductor社の「LatticeMico8」は，必要最小限の機能だけを少ない論理規模で実現したCPUです．30年近く前に米国AMD社がAm2901というビットスライス・マイコンを出荷していたのを思い出します．VHDLとVerilog HDLの両方の記述が無償で提供されており，Lattice社のWebサイト（http://www.latticesemi.co.jp/）からダウンロードできます．

LatticeMico32は，周辺機能と組み合わせて活用する32ビットのCPUコアです．VHDLとVerilog HDLの両方の記述が無償で提供されています．

2. FPGAにCPUコアを搭載する意義

FPGAにCPUを搭載する意義はどこにあるのでしょうか．筆者は以下の3点に集約されると考えています[(1)]．

- 使わない機能のことを考えずにすむ
- 1個からSOC（system on a chip）が作れる
- 機能/性能のトレードオフ

● 使わない機能のことを考えずにすむ

汎用マイコンは，さまざまなアプリケーションに対応するため，必要と思われるすべての機能を搭載しています．CPU

の機能は当然として，周辺機能としてUART（universal asynchronous receiver transmitter），A-Dコンバータ，D-Aコンバータ，タイマ，汎用I/Oなどが1～複数個搭載されています．設計者は，自分が設計しようとするアプリケーションの実現に必要な機能がいちばん多く含まれている品種を選択します．このとき，使わない機能についてはあまり意識しません．しかしよく考えてみると，ユーザはその使わない部分についてもお金を払っているのです．

さらにソフトウェア開発についても同様です．多くの機能を限られたピン数のパッケージに入れるために，一つのピンに複数の機能が割り当てられることが少なくありません．このため，ピンにどの機能を割り当てるのかを設定する初期化プログラムが必要です．機能とピンが直結していれば，初期化プログラムが簡単になります．

動作パラメータも多くのアプリケーションではソフトウェアによる初期設定が不要なはずです．固定値でよいのであれば，ハードウェア設計時に組み込んでしまえるはずだからです．UARTの通信速度などはこれに相当します．

● 1個からSOCを作れる

LSI製造プロセス技術の微細化に伴い，ASICの開発費（NRE：non recurring expenses）は1億円を超えることがあります．その費用をLSIの総生産数で回収しようとすると，1ヵ月に10万個規模で販売可能でないと開発できないと言われています．

ほんとうはカスタムLSIが欲しくても，単独で開発費を捻出できないケースは増えています．たとえ開発費を捻出できたとしても，開発期間やライフタイムが製品計画と合わないこともあります．このため，汎用CPUやASSPから近い物を選択し，足りない部分をFPGAなどで補ってシステムを開発せざるを得なくなっています．

セット・メーカにとっての目的は，売れる製品をいち早く市場投入することであって，LSIを作ることではありません．そこで「可能な限り高速なCPUを積んで，欲しい周辺モジュールを一つでも多く搭載したSOC」というのが部品選択の最重要ポイントになっているのです．

一方，FPGAは自由に回路を構成できる，ミニマム・オーダ1個のカスタムLSIです．開発コストを除いた部品単価は，ASICと比べてFPGAのほうが高価です．ただし，低コストFPGAでは差は縮まってきており，家電製品への採用実績も増えています．ほんとうに欲しい機能だけを組み込んだSOCを自分で作れるのは，FPGAだけの特権です．

● 機能や性能のトレードオフ

機能や性能のトレードオフを考慮しながら，ハードウェアとソフトウェアのどちらで実現するのかについて，柔軟に選択できます．

「論理」というと，ハードウェアととらえがちです．しかし実は論理的動作をするものの総称であり，実現手法は含まれていません．つまり目的とする動作を実現するのであれば，ハードウェア，ソフトウェアのどちらか，もしくは両方の手法を使ってもかまわないわけです．そこで筆者は次のステップを推奨しています．

（1）極力ソフトウェアで実現する

制御対象がFPGAの外部に接続されているとしましょう．そのインターフェース仕様は公開されているものの，手元にそれをサポートした周辺モジュールがなく，そのインターフェース仕様を知っているハードウェア設計者もいません．このような場合，まずは汎用I/Oポートを使ってすべての端子を接続してしまいます．そして，すべてソフトウェアによってI/Oポートを制御してみます．こうして対象デバイスの制御方法を実際に動作させながら把握します．

ソフトウェアなので高速動作は期待しません．低速なデバイスではこのまま実回路に適用してもまったく問題ありませんが，ここではあくまでも機能的な確認が目的です．

（2）CPUで負荷が重い機能からハードウェア化する

実際に制御対象のデバイスが動いているわけですから，ソフトウェアで書かれた論理を忠実にハードウェア化すればよいことになります．制御のために書いたソフトウェアから，レジスタの構成などを決定し，動作する論理回路を設計していきます．ソフトウェアの構成をうまく設計しておけば，アクセスする関数の内部を書き替えるだけで，それ以外はまったく変更せずにCPU負荷の低減，および処理の高速化が可能となります．

今回はFPGA外部に制御対象がある場合を例にしましたが，FPGA内部でも考えかたは同じです．例えばCPUがサポートしていない高度な関数演算などをアクセラレータとして実現することがこれにあたります．演算アルゴリズムさえ決まっていれば，ソフトウェアを見ながら高速動作する演算回路を作ることは難しくはありません．

3. LSI設計におけるC言語

カスタムLSIの世界では何年か前から，HDL（hardware description language）によるRTL（register transfer level）設計の次は，C言語による動作合成だと言われています．SystemC（http://www.systemc.org/）やSpecC（http://www.specc.gr.jp/）に代表される言語仕様が策定され，これらに対応した開発ツールが発表されています．最近では，純粋なソフトウェア技術者が短期間でASICを開発できたという事例も紹介されていますが[3]，まだ幅広い普及には至っていません．それはなぜでしょうか．

● C言語によるハードウェア設計のメリット

ハードウェア設計にC言語設計を取り込むことの最大のメリットは，論理設計のための専用言語（HDL）を覚える必要がないということです．組み込みシステムだけでなく，パソコンも含めたCPU搭載機器のソフトウェアのほとんどは，C言語で開発されています．アプリケーションやミドルウェアに関しては，ソフトウェア技術者のほうが理解している場合が多いようです．数十万人はいるというソフトウェア技術者がハードウェア設計リソースにもなれるとしたら，すばらしいと思いませんか．C言語によるハードウェア設計は，LSI設計に対するハードルの一つを取り除くものと考えられています．

第2のメリットは，言語によるハードウェア設計より歴史のあるソフトウェア開発環境でデバッグができるということです．設計者が使い慣れた言語とツールを用いてデバッグできれば，品質向上にも貢献するはずです．

例えば，画像処理におけるアルゴリズムの検証では，入力データとして実際の画像を用いることがいちばん確実です．最近のパソコンは，カメラ接続用のインターフェースや，複数の画像フォーマットを取り扱えるエディタやビューワが装備されているので，それらを有効利用するのがあたりまえになっています．すると開発言語は当然C/C++になります．パソコンでアルゴリズムが確定しても，最終ターゲットがLSIの場合，そのアルゴリズムをどうやってハードウェア化するかを考える必要があります．

① C言語からの変換ツールを入手する．
② C言語からHDLにコーディングし直す．
③ DSP機能を持つCPU向けにソフトウェアを修正する．

①がもっとも効率的です．③は，どうしてもハードウェア化できない場合にとられる手法にすぎません．

● C言語によるハードウェア設計のデメリット

ハードウェア設計者とC言語設計の話をすると，「C言語がソフトウェア開発言語として普及していることは十分理解している．しかしレジスタやクロックという概念のないC言語で，ハードウェアを記述するなんて考えられない．結局コンパイラ依存になり，コンパイラの性能がハードウェアの品質を左右することになる．記述に制約をかけたC言語をHDLの代わりとして使用するだけならば，HDLだけで十分だ」という意見が多く聞かれます．

LSI開発は，テープアウトで設計者の手元を離れると，動作しているようすを目にすることが少ないしごとです[2]．LSI設計者はシミュレーション結果を信じるしかありません．出荷後のオンライン・アップデートもできません．何度も修正を繰り返せるソフトウェア開発と同じ土俵で語ってしまうと，痛い目に遭うでしょう．

1

● Cベース言語の使いどころ

現時点ではC言語設計はテストベクタを中心とする検証環境に留まり，HDLによる論理設計が主流のままであると筆者は考えています．

しかし，ツールの進化は目を見張るものがあります．現状の課題は徐々にクリアされていくことはまちがいありません．Cベース言語からの動作合成技術がさらに発展し，性能や品質，開発期間などの面においてHDLベースのRTL設計を上回る成果が出始めたら，現役のハードウェア技術者も積極的に導入し，一気に普及することでしょう．

すべてのカスタム論理をFPGAで実現し，ハードウェアも自由にアップデートできるシステムが標準になれば，ハードウェアとソフトウェアの境界がなくなるのですが．それは夢物語でしょうか．

参考・引用*文献

(1) 浅井 剛；ソフト・マクロのCPUを使おう！，Design Wave Magazine，2003年8月号，pp.20-27.
(2) 浅井 剛；ようこそFPGA/ASIC設計の世界へ！，Design Wave Magazine，2005年5月号，pp.22-29.
(3) たった1人のソフト技術者がMPEG-4チップを半年で開発，http://techon.nikkeibp.co.jp/NEWS/dac2003/020729_02_1.html

あさい・たけし　東京計器工業（株）FPGAコンソーシアム　監事